LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-TR-680499

Multigrid Reduction in Time for Nonlinear Parabolic Problems

J. B. Schroder, R. D. Falgout, T. Manteuffel, B. O'Neill

January 4, 2016

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Multigrid Reduction in Time for Nonlinear Parabolic Problems

R.D. Falgout * T.A. Manteuffel † B. O'Neill † J.B. Schroder *

January 5, 2016

Abstract

The need for parallel-in-time is being driven by changes in computer architectures, where future speed-ups will be available through greater concurrency, but not faster clock speeds, which are stagnant. This leads to a bottleneck for sequential time marching schemes, because they lack parallelism in the time dimension. Multigrid Reduction in Time (MGRIT) is an iterative procedure that allows for temporal parallelism by utilizing multigrid reduction techniques and a multilevel hierarchy of coarse time grids. MGRIT has been shown to be effective for linear problems, with speedups of up to 50 times.

The goal of this work is the efficient solution of nonlinear problems with MGRIT, where efficient is defined as achieving similar performance when compared to a corresponding linear problem. As our benchmark, we use the p-Laplacian, where $p = 4$ corresponds to a well-known nonlinear diffusion equation and $p = 2$ corresponds to our benchmark linear diffusion problem. When considering linear problems and implicit methods, the use of optimal spatial solvers such as spatial multigrid imply that the cost of one time step evaluation is fixed across temporal levels, which have a large variation in time step sizes. This is not the case for nonlinear problems, where the work required increases dramatically on coarser time grids, where relatively large time steps lead to worse conditioned nonlinear solves and increased nonlinear iteration counts per time step evaluation. This is the key difficulty explored by this paper. We show that by using a variety of strategies, most importantly, spatial coarsening and an alternate initial guess to the nonlinear time-step solver, we can reduce the work per time step evaluation over all temporal levels to a range similar with the corresponding linear problem. This allows for parallel scaling behavior comparable to the corresponding linear problem.

1 Introduction

Previously, ever increasing clock speeds allowed for the speed-up of sequential time integration simulations of a fixed size and also for stable runtimes (wall clock times) for simulations that are refined in space (and usually time). However, clock speeds are now stagnant, leading to the sequential time integration bottleneck. Future increases in compute power will be available from more concurrency, and hence speedups for time marching simulations must also come from increased concurrency.

The simplest example of the sequential bottleneck is when the communication cost associated with adding a processor in space outweighs the added computational power, i.e., the code has exhausted spatial parallelism. Moreover, refinements in space usually lead to refinements in time. If the runtime required for each time step is stagnant or increasing, this will only lead to ever longer simulation times. By allowing for parallelism in time, much greater computational resources can be brought to bear, and overall speedups can be achieved. To this end, interest in parallel-in-time methods has grown over the last decade. Perhaps the most well known parallel in time time algorithm, Parareal [23], is equivalent [13] to a two-level multigrid scheme. We focus on the multigrid reduction in time (MGRIT) method [10]. MGRIT is a true multilevel algorithm and has optimal parallel communication behavior, as opposed to a two-level scheme where the size of the coarse-level limits concurrency.

Work on parallel-in-time methods actually goes back at least 50 years [30] and includes a variety of approaches. Work regarding direct methods includes [29, 32, 25, 6, 14]. There are iterative approaches,

*Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, P.O. Box 808, L-561, Livermore,CA 94551. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. LLNL-TR-680499

†Department of Applied Mathematics, University of Colorado at Boulder, Boulder, Colorado

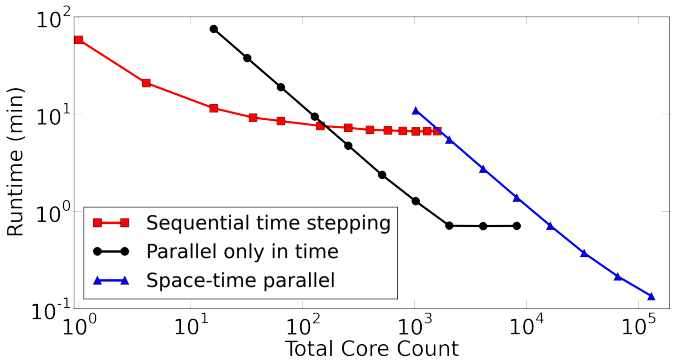

Figure 1: Time to solve $2D$ linear diffusion on a $(128)^2 \times 16385$ space-time grid using sequential time stepping and two processor decompositions of MGRIT. [10]

as well, based on multiple shooting, domain decomposition, waveform relaxation, and multigrid including [21, 15, 24, 1, 16, 17, 34, 5, 35, 33, 19, 18, 23, 7, 28, 9, 36, 10]. For a gentle introduction to this history, please see the review paper [12]. Our work focuses on multigrid approaches (and MGRIT in particular) because of multigrid's optimal algorithmic scaling for both parallel communication and number of operations. We are additionally attracted to MGRIT because of its non-intrusive nature, where the user wraps an existing sequential time-stepper according to the interface of our MGRIT implementation.

The goal is to solve a general first order ordinary differential equation (ODE) and corresponding time discretization:

$$u_t = f(u,t),\ u(0) = u_0,\ t \in [0,T], \tag{1}$$

$$u(t+\delta t) = \Phi(u(t), u(t+\delta t)) + g(t+\delta t), \tag{2}$$

where Φ is a nonlinear operator that encapsulates the chosen time stepping routine and g incorporates all solution independent terms. The application of Φ is either a matrix vector multiplication, e.g. forward Euler, or a spatial solve, e.g. backward Euler.

Classical time marching schemes are optimal in that they move from time $t = 0$ to $t = T$ with the fewest possible applications of Φ. Applying Φ iteratively, in comparably expensive, but highly parallel, multigrid cycles, MGRIT sacrifices efficiency for temporal concurrency. Both methods are optimal, i.e. $O(N)$, but the constant for MGRIT is higher. This creates a crossover point wherein, the added concurrency accounts for the extra computational work. Beyond this crossover MGRIT provides speedup over sequential methods.

Application of MGRIT to linear parabolic problems was studied extensively in [10]. Figure 1 shows a strong scaling study of MGRIT for linear diffusion on the machine Vulcan, an IBM BGQ machine at Lawrence Livermore National Laboratory. The problem size is $(257)^2 \times 16385$ and three data sets are presented, standard sequential time stepping, time-only parallelism run of MGRIT and a space-time parallel run of MGRIT. The space-time parallel runs used an 8×8 processor grid in space, with all additional processors being added in time. Both MGRIT curves represent the use of temporal and spatial coarsening, so that the ratio of dt/dx^2 is fixed on coarse time-grids. The maximum speedup achieved by the blue curve is approximately 50, and the crossover point where MGRIT provides a speedup is at about 128 processors in time. Our goal is to make the overall performance (i.e., crossover point and speedup) of MGRIT for nonlinear problems similar to that for linear problems.

When considering the performance of MGRIT, the application of Φ is the dominant process. For a linear problem with implicit time stepping each application of Φ equates to solving one linear system. When an optimal spatial solver such as classical spatial multigrid [26, 4, 31, 20] is used, the work required for a time step evaluation is constant across all time levels (and associated time step sizes). However when Φ becomes nonlinear, each application of Φ becomes an iterative nonlinear solve, whose conditioning usually depends on the time step size. To explore the effects of introducing a nonlinearity, we consider the model nonlinear parabolic problem known as the p-Laplacian,

$$u_t(\mathbf{x},t) - \nabla \cdot (|\nabla u(\mathbf{x},t)|^{p-2} \nabla u(\mathbf{x},t)) = b(\mathbf{x},t),\ \mathbf{x} \in \Omega,\ t \in [0,T] \tag{3}$$

subject to an initial condition and Neumann boundary conditions

$$|\nabla u(\mathbf{x},t)|^{p-2}\nabla u(\mathbf{x},t)\cdot\mathbf{n} = g(\mathbf{x},t),\ \mathbf{x}\in\partial\Omega,\ t\in(0,T] \tag{4}$$

$$u(\mathbf{x},0) = u_0(\mathbf{x}), \mathbf{x}\in\Omega. \tag{5}$$

The p-Laplacian for $p = 4$ is well-known as a means of modeling soil erosion and transport [2] and has also found uses in image processing (denoising, segmentation and inpainting) and machine learning, see [8] for an overview and [22] for a gentle introduction. In this paper, our model nonlinear problem corresponds to $p = 4$, while the comparable linear problem corresponds to $p = 2$, which is the standard Laplace operator.

A naive application of MGRIT to (3) shows large increased nonlinear iteration counts (here we use Newton's method) per Φ evaluation on the coarser temporal grids because of the relatively large time steps (compared to the finest grid) and the associated poor initial guess to our Newton solver. These increases counteract the strength of multigrid, where speedup is achieved by using cheap coarse grid problems to accelerate convergence on the fine grid. Therefore the average number of Newton iterations per time step on a per level basis is our choice of *heuristic*. The goal is to minimize the heuristic so that we can achieve similar efficiencies for our model nonlinear problem and it's comparable linear problem.

Towards the goal of minimizing the number of Newton iterations per Φ evaluation, we pursue a spatial coarsening strategy of keeping $\delta t/\delta x^2$ fixed on all time grids. This keeps the conditioning of Φ fixed over all time levels and ultimately requires far fewer Newton iterations per Φ evaluation. In addition, the smaller problem sizes drastically reduce coarse grid compute times. MGRIT semi-coarsens in time and is agnostic to the spatial discretization, thus the use of the spatial coarsening option requires the user implementation of spatial restriction and interpolation operators.

However, introducing spatial coarsening can degrade MGRIT convergence and, in some cases, result in non-convergence. Analysis suggests this occurs because MGRIT with spatial coarsening no longer corresponds to an exact reduction method with a modified coarse-grid operator. Essentially, the spatial prolongation and restriction operators have a null space which is invisible to the coarse grid correction. Motivated by the dramatic computational savings, but tempered by the deleterious effects on convergence caused by spatial coarsening, we choose to do a *delayed* spatial coarsening strategy, determined experimentally, that begins spatial coarsening on the second coarse-grid.

To further reduce the average number of Newton iterations per Φ evaluation, we also investigate an improved initial guess. Classically the best initial guess for the Newton solver is the previous time step. However for large time steps, this initial guess is poor, resulting in large increases in Newton iterations per Φ evaluation when moving to coarse time grids. The iterative nature of MGRIT gives us another option. We instead use the approximate solution at the corresponding time point from the previous MGRIT iteration as the initial guess and this reduces the average number of Newton iterations, per Φ evaluation, to below that of an equivalent sequential time integration.

Two more important strategies include a progressive loosening of the Newton solve tolerance on coarser levels and avoiding unnecessary work on the first MGRIT cycle. During the first cycle while traversing down the temporal grids, there is no useful information at later time steps, except the initial guess, which makes the relaxation process ineffective, expensive and not useful. We therefore skip it. The last two strategies employed are optimizing the number of levels in the hierarchy and a further loosening of the Newton solver tolerance during the first three MGRIT iterations on all levels, while the approximate solution is still poor.

The most important strategies are spatial coarsening and the improved initial guess, while the other strategies combined make a similarly big impact on runtime

In Section 2, we discuss the general MGRIT framework. In Section 3, some implementation details are given such as our choice of Newton's method as our nonlinear time-stepper. In Section 4, our heuristic, the average number of Newton iterations per time step, is proposed and justified. In Section 5, we use our heuristic to investigate our strategies for making MGRIT more efficient. In sections 6.1 and 6.2, we present multigrid and strong scaling results. The multigrid scaling study shows optimal MGRIT iterations for a domain refinement study, bounded independently of problem size. The strong scaling results show scaling behavior similar to that of the comparable linear problem.

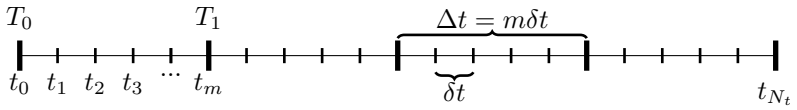

Figure 2: Fine- and coarse-grid temporal meshes. Fine-grid points are present on only the fine-grid, whereas coarse-grid points are on both the fine- and coarse-grid.

2 MGRIT overview

We now give a brief overview of the MGRIT algorithm. Define a uniform temporal grid with time step δt and nodes $t_j,\ j = 0, \ldots, N_t$ (non-uniform grids are also supported). Further define a coarse temporal grid with time step $\Delta T = m\delta t$ and nodes $T_j = j\Delta T,\ j = 0, 1, \ldots, N_t/m$ for some coarsening factor m. This is depicted in Figure 2. In block triangular form, the time-stepping problem (2) is

$$A(\mathbf{u}) = \begin{bmatrix} I & & & \\ -\Phi_0 & I & & \\ & \ddots & \ddots & \\ & & -\Phi_{N_t-1} & I \end{bmatrix} \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_{N_t} \end{bmatrix} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_{N_t} \end{bmatrix} = \mathbf{g}. \tag{6}$$

Classical time marching is a forward block solve of this system. MGRIT solves this system iteratively, in parallel, using a coarse-grid correction scheme based on multigrid reduction. Both are $O(N)$ methods but MGRIT is highly concurrent. Multigrid reduction strategies are essentially approximate cyclic reduction methods and as such, successively eliminate unknowns in the system. If the fine points are eliminated, we obtain the system

$$A_\Delta(\mathbf{u}_\Delta) = \begin{bmatrix} I & & & \\ -\Phi^m & I & & \\ & \ddots & & \\ & & -\Phi^m & I \end{bmatrix} \begin{bmatrix} \mathbf{u}_{\Delta,0} \\ \mathbf{u}_{\Delta,1} \\ \vdots \\ \mathbf{u}_{\Delta,N_t} \end{bmatrix} = R\mathbf{g} = \mathbf{g}_\Delta. \tag{7}$$

By definition defining "ideal" restriction R and interpolation P, we can alternately obtain this system in a multigrid fashion. Let,

$$R = \begin{bmatrix} I & & & & & & & & & \\ & \Phi^{m-1} & \ldots & \Phi & I & & & & & \\ & & & & & \ddots & & & & \\ & & & & & & \Phi^{m-1} & \ldots & \Phi & I \end{bmatrix}, \tag{8a}$$

$$P = \begin{bmatrix} I & \Phi^T & \ldots & \Phi^{m-1,T} & & & & \\ & & \ddots & & & & & \\ & & & & I & \Phi^T & \ldots & \Phi^{m-1,T} \end{bmatrix}^T. \tag{8b}$$

The interpolation injects at coarse points and harmonically extends values at coarse points to fine points, i.e., it is injection from the coarse- to fine-grid followed by F-relaxation (defined below). In a similar fashion, restriction is F-relaxation followed by injection from the fine to coarse-grid.

With this, the "ideal" coarse-grid operator is $A_\Delta = RAP$. We refer to this as the ideal because the solution of (7) yields an exact solution at the coarse points. This coarse grid is essentially a compressed version of the problem. If this is followed by interpolation, then the exact solution is also available at fine points. The limitation of this exact reduction method is that the coarse-grid problem is in general as expensive to solve as the original fine-grid problem (because of the Φ^m evaluations). Multigrid reduction addresses this by approximating A_Δ with B_Δ through the use of an approximate coarse-grid time-stepper

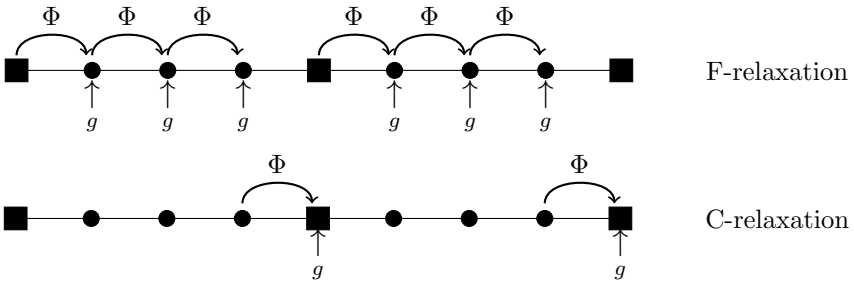

Figure 3: F- and C-relaxation for a coarsening by factor of 4

Φ_Δ,

$$B_\Delta = \begin{bmatrix} I & & & \\ -\Phi_\Delta & I & & \\ & \ddots & & \\ & & -\Phi_\Delta & I \end{bmatrix}. \tag{9}$$

One obvious choice for defining Φ_Δ is to re-discretize the problem on the coarse grid so that a coarse-grid time step is roughly as expensive as a fine-grid time step. This is the choice that we make. For instance with backward Euler, one simply uses a larger time step size. Convergence of MGRIT is governed by the approximation $A_\Delta \approx B_\Delta$ and this choice of using a re-discretization of Φ with $\Delta T = m\delta t$ has proven effective, [10], [11]. Last, the definition of this algorithm relies upon Φ and Φ_Δ, but the internals of these functions need not be known. This is the non-intrusive aspect of MGRIT. The user defines the time-stepper and can wrap existing codes to work with the MGRIT framework.

Working in conjunction with the coarse-grid problem, is relaxation. The coarse-grid is used to compute an error correction based on the residual equation (see Algorithm 1), while relaxation is a local fine-grid process used to resolve fine-scale behavior. Figure 3 shows the actions of F- and C-relaxation on a temporal grid with $m = 4$. F-relaxation propagates the solution from each coarse point to the neighboring F-points to the right (i.e., forward in time). Overall, relaxation is highly parallel. Each interval of F-points can be updated independently during F-relaxation and each C-point update during C-relaxation is similarly independent.

2.1 Schur-Complement perspective for linear problems

If we assume for simplicity that Φ is linear, then we can derive the method as an approximate Schur-complement approach with relaxation, i.e., a two-level multigrid method. This derivation is meant to give the reader better intuition into how the method works. Grouping together fine (f) and coarse (c) points, we can write the re-ordered system as

$$\begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix} \begin{bmatrix} \mathbf{u}_f \\ \mathbf{u}_c \end{bmatrix} = \begin{bmatrix} \mathbf{g}_f \\ \mathbf{g}_c \end{bmatrix}.$$

A simple Schur-complement decomposition gives

$$\begin{bmatrix} I_f & 0 \\ A_{cf}A_{ff}^{-1} & I_c \end{bmatrix} \begin{bmatrix} A_{ff} & 0 \\ 0 & A_{cc} - A_{cf}A_{ff}^{-1}A_{fc} \end{bmatrix} \begin{bmatrix} I_f & A_{ff}^{-1}A_{fc} \\ 0 & I_c \end{bmatrix} \begin{bmatrix} \mathbf{u}_f \\ \mathbf{u}_c \end{bmatrix} = \begin{bmatrix} \mathbf{g}_f \\ \mathbf{g}_c \end{bmatrix},$$

This decomposition naturally implies operators R and P, known as ideal restriction and interpolation (equivalent to (8)), and S as

$$R = \begin{bmatrix} -A_{fc}A_{ff}^{-1} & I_c \end{bmatrix}, \; P = \begin{bmatrix} -A_{ff}^{-1}A_{fc} \\ I_c \end{bmatrix}, \; S = \begin{bmatrix} I_f \\ 0 \end{bmatrix}. \tag{10}$$

Noting that $S^TAS = A_{ff}$ and $RAP = A_{cc} - A_{cf}A_{ff}^{-1}A_{fc}$, we have

$$A^{-1} = P(RAP)^{-1}R + S(S^TAS)^{-1}S^T.$$

This gives the multiplicative identity,

$$0 = I - A^{-1}A = (I - P(RAP)^{-1}RA)(I - S(S^TAS)^{-1}S^TA), \tag{11}$$

The first term corresponds to the error propagator of coarse grid correction using the ideal Petrov-Galerkin coarse grid operator, RAP, and the second, the error propagator for F-relaxation.

To produce an iterative multigrid method, multigrid reduction methods commonly use approximations to the "ideal" coarse-grid operator (here, substitute B_Δ for A_Δ) along with ideal interpolation. With this, the two grid error propagator is defined as

$$(I - PB_\Delta^{-1}RA)(I - S(S^TAS)^{-1}S^TA) = P(I - B_\Delta^{-1}A_\Delta)R_I, \tag{12}$$

where $R_I = [0, I_c]^T$ is the injection operator. It was shown in [10] that, for an optimal scaling algorithm, F-relaxation must be replaced by FCF-relaxation. The corresponding error propagator for FCF-relaxation is $P(I - A_\Delta)R_I$. With this, the final two grid error propagation operator is

$$(I - PB_\Delta^{-1}RA)(P(I - A_\Delta)R_I) = P(I - B_\Delta^{-1}A_\Delta)(I - A_\Delta)R_I. \tag{13}$$

2.2 MGRIT algorithm for nonlinear problems

Putting the above components together, we describe the MGRIT algorithm for nonlinear problems. This is a straight-forward extension of the linear algorithm [10] using the FAS (nonlinear multigrid) scheme [3]. This is a reasonable approach to extending the algorithm to nonlinear problems. In particular, the F-relaxation, two-grid variant is equivalent [13] to the popular Parareal algorithm, which has been shown effective for a variety of nonlinear problems. The FAS description of MGRIT first appeared in [11].

For efficiency, we note that $RAP = R_IAP$, where R_I is injection at the coarse points. Thus, we use injection to map to the coarse level (like Parareal). The exception is the *spatial coarsening* option where spatial restriction and interpolation functions, $R_x()$ and $P_x()$, are used to coarsen in space as well as time. With this, the MGRIT algorithm is presented in Algorithm 1 as a two-level method, but can be used in a multilevel setting by recursively applying the algorithm at Step 3.

Algorithm 1 MGRIT(A, $\mathbf{u}$, $\mathbf{g}$)

1: Apply F- or FCF-relaxation to $A(\mathbf{u}) = \mathbf{g}$.
2: Inject the fine grid approximation and its residual to the coarse grid
$u_{\Delta,i} \leftarrow u_{mi}, \quad r_{\Delta,i} \leftarrow g_{mi} - (A(\mathbf{u}))_{mi}$
3: If Spatial coarsening then
$u_{\Delta,i} \leftarrow R_x(u_{\Delta,i}), \quad r_{\Delta,i} \leftarrow R_x(r_{\Delta,i})$
4: Solve $B_\Delta(\mathbf{v}_\Delta) = B_\Delta(\mathbf{u}_\Delta) + \mathbf{r}_\Delta$.
5: Compute the coarse grid error approximation: $\mathbf{e}_\Delta = \mathbf{v}_\Delta - \mathbf{u}_\Delta$
6: If Spatial coarsening then
$e_{\Delta,i} \leftarrow P_x(e_{\Delta,i})$,
7: Correct using ideal interpolation: $\mathbf{u} = \mathbf{u} + P\mathbf{e}_\Delta$

The reader will note that with exact arithmetic, MGRIT with FCF-relaxation propagates the initial condition two full coarse grid time intervals ($2\Delta t$) each cycle. Thus, MGRIT is equivalent to a sequential direct solve in $N_t/(2m)$ iterations. With F-relaxation only, the sequential solution is achieved in N_t/m iterations. The speedup comes from the fact that the method converges in $O(1)$ iterations.

A variety of cycling strategies are available in multigrid (e.g., V, W, F). We use the standard V-cycle depicted in Figure 4. This corresponds to Algorithm 1 with the "Solve" step 3 turned into a single recursive call. The recursion ends when a trivially sized grid of say 5 time points is reached, at which point a sequential solver is used. On the left, we see the option of coarsening in space and time such that the temporal resolution δt and the spatial resolution δx are fixed according to the "parabolic" ratio of $\delta t/\delta x^2$ fixed. On the right, we see a time-only coarsening V-cycle.

Our chosen implementation of MGRIT is XBraid [37], an open source package developed at LLNL. XBraid conforms to MGRIT's non-intrusive philosophy and requires the user to wrap an existing time-stepping routine, as well as define a few other basic operations like a state-vector norm and inner-product.

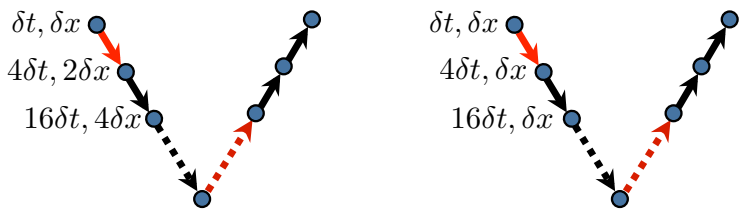

Figure 4: Example multigrid V-cycle with space-time coarsening that maintains $\delta t/\delta x^2$ fixed on the left, and then time-only coarsening on the right.

The key computational kernel is the time-stepping (or Φ routine), but all the specifics are opaque to XBraid and done in user code. This allows the user to add temporal parallelism to existing time stepping codes with minimal modifications. For more details, see [10] and [37].

3 Model problem implementation

We discretize (3) with a finite element space V, yielding the weak form

$$\langle u_t, v\rangle + \langle |\nabla u|^{p-2}\nabla u, \nabla v\rangle = \langle b, v\rangle + \langle g, v\rangle_{\partial\Omega},\ \forall v \in V. \tag{14}$$

Discretizing using a backward Euler method and the temporal mesh in Figure 2, we have

$$\left\langle \frac{u_{k+1}-u_k}{\delta t}, v \right\rangle + \langle |\nabla u_{k+1}|^{p-2}\nabla u_{k+1}, \nabla v\rangle = \langle b_{k+1}, v\rangle + \langle g_{k+1}, v\rangle_{\partial\Omega},\ \forall v \in V,\ k = 0, 1, \ldots, N_t - 1. \tag{15}$$

Defining

$$\hat{\Phi}(u)(v) = \langle u, v\rangle + \delta t\langle |\nabla u|^{p-2}\nabla u, \nabla v\rangle, \tag{16}$$

$$f_k(v) = \langle u_k - b_{k+1}\delta t, v\rangle - \delta t\langle g_{k+1}, v\rangle_{\partial\Omega}, \tag{17}$$

we have

$$\hat{\Phi}(u_{k+1})(v) = f_k(v),\ \forall v \in V,\ k = 0, 1, 2, \ldots, N_t - 1. \tag{18}$$

Each time step corresponds to the solution of this nonlinear system.

All tests were completed with $T = 4s$ and $\Omega = [0, 2]^2$ for a regular grid. The forcing function, $b(\mathbf{x}, t)$, was chosen such that the exact solution of the problem is

$$u(x, y) = \sin(\kappa x)\sin(\kappa y)\sin(\tau t),$$

where $\kappa = \pi$ and $\tau = (2 + 1/6)\pi$. Unless otherwise stated, we use the p-Laplacian with $p = 4$. The spatial discretization is computed with standard linear quadrilateral elements with MFEM [27], a parallel finite element code.

We use Newton's method to solve for each individual time step (18), i.e.,

$$u^{j+1} = u^j - \hat{\Phi}'(u^j)(v)^{-1}[\hat{\Phi}(u^j)(v) - f(v)] \tag{19}$$

$$= u^j - \delta u^j. \tag{20}$$

where we dropped the temporal subscript for clarity and

$$\hat{\Phi}'(u)(v)[w] = \lim_{a\to 0}\frac{\hat{\Phi}(u - aw)(v) - \hat{\Phi}(u)(v)}{a}, \tag{21}$$

$$= \langle w, v\rangle + \delta t\langle \left[|\nabla u|^{p-2} + (p-2)(\nabla u)(\nabla u)^T\right]\nabla w, \nabla v\rangle. \tag{22}$$

is the Fréchet derivative.

δt	δx	a_ℓ	Max	Min
1/1024	1/64	3.5	4	2
1/256	1/64	4.1	5	2
1/64	1/32	7.1	11	3
1/16	1/16	8.6	16	4
1/4	1/4	11.0	16	5
1	1	9.5	12	5

(a) Spatial coarsening (as used here)

δt	δx	a_ℓ	Max	Min
1/1024	1/64	3.5	4	2
1/256	1/64	4.1	5	2
1/64	1/64	9.5	14	3
1/16	1/64	12.3	21	4
1/4	1/64	13.7	19	8
1	1/64	13.8	18	7

(b) No spatial Coarsening

Table 1: Baseline iteration counts for Newton solver using the sequential method.

4 Heuristic for efficient MGRIT

4.1 Numerical parameters

The numerical testing parameters (unless otherwise mentioned) are as follows. The Newton tolerance is fixed at 10^{-7}. The optimal spatial solver for each Newton iteration is BoomerAMG [20] and uses the following parameter setting: HMIS coarsening (coarsen-type 10), one level of aggressive coarsening, symmetric L1 Gauss-Seidel (relax-type 8), extended classical modified interpolation (interp-type 6), and interpolation truncation equal to 4 nonzeros per row. The machine used for all numerical tests (including the scaling studies) is Vulcan, an IBM BG/Q machine at Lawrence Livermore National Laboratory.

Our test problem size (until the scaling studies) is a $(64)^2 \times 4096$ space-time grid on the domain $[0,2]^2 \times [0,4]$ using 1 processor in space and 64 processors in time. MGRIT is always used with V-cycles and FCF-relaxation and a fixed stopping criteria of $10^{-9}/(\sqrt{\delta t}\delta x)$, so that the same tolerance, relative to resolution, is used in all cases. This is an overly tight tolerance with respect to discretization error, set in large part because we want to investigate the algorithm's asymptotic convergence properties. The coarsening factor is $m = 4$.

4.2 Sequential time-stepping baseline for efficiency

The establish a baseline for efficiency, we run a few experiments with classical sequential time-stepping, using the previous time-step value as the initial guess at each subsequent time-step. Table 1 gives the average, maximum and minimum Newton solver iteration counts over the selected time step sizes. On the left we mimic the grids used by MGRIT when applying spatial coarsening (note that spatial coarsening is delayed). On the right we show sequential solver performance for grids associated with no spatial coarsening. We set as our realistic goal to make the a_ℓ from our eventual MGRIT solver match the numbers in Table 1, e.g., if level 3 in an MGRIT hierarchy has $\delta t = 1/64$ and $\delta x = 1/32$, we target MGRIT taking on average 7.1 Newton iterations per time step. The maximum and minimum values are also important because we uniformly distribute points in time in parallel. Thus, if the maximum is much higher than the average, then this will represent an inefficiency in MGRIT. Unfortunately, XBraid does not currently support load balancing. Last, these baseline results make it clear that the number of Newton iterations required for convergence is highly dependent on the grid, with there being an advantage to coarsening simultaneously in space and time.

4.3 Heuristic for naive MGRIT

We now discuss our heuristic for finding an efficient MGRIT for the model problem. We choose a_ℓ, or the average number of Newton iterations per time step on level ℓ. To understand our heuristic in a broad sense, let a be the average over all levels. In [10] it was shown that, for a linear problem, the computational model for MGRIT's cost, c_{linear}, is essentially defined by the number of calls to the linear solve routine in Φ,

$$c_{linear} \propto \nu_t(2m/(m-1)+1)N_t, \tag{23}$$

Iteration	$\delta t = 1/1024$	1/256	1/64	1/16	1/4	1
0	6.67	6.66	8.46	14.2	17.1	16.5
1	3.67	4.28	8.41	12.3	16.0	18.0
2	3.47	4.13	8.36	12.8	16.8	19.0
3	3.45	4.06	8.47	12.9	16.3	19.2
4	3.45	4.06	8.43	12.8	16.3	19.2
5	3.45	4.06	8.44	12.9	16.3	19.2

Table 2: The average number of Newton iterations per time step (a_ℓ) across each temporal level and MGRIT iteration

Iteration	$\delta t = 1/1024$	1/256	1/64	1/16	1/4	1	Total (assuming $m = 4$)
0	$13.3m$	13.3	16.9	28.4	34.2	33.0	179.0
1	$7.34m$	8.56	16.8	24.6	32.0	36.0	147.3
2	$6.90m$	8.24	16.7	25.6	33.6	38.0	149.7
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 3: For processes active on each temporal level, estimated average number of Newton iterations to carry out FCF-relaxation, based on Table 2

where ν_t is the number of MGRIT iterations. In a nonlinear setting (assuming one linear solver per Newton iteration) this becomes

$$c_{nonlinear} \propto a\, c_{linear}. \tag{24}$$

Minimizing a is key to minimizing the required computational work. However, we will see that all computational work is not "equal", with the work on coarse-grids being the chief target for reduction.

To understand why our heuristic choice must be per level (and not just a), we apply MGRIT in a naive fashion, as one might apply MGRIT initially to an existing sequential implementation of the model problem. Here, the experiments always use the previous time step as the initial guess to the Newton solver and there is no spatial coarsening.

Table 2 gives the average number of Newton iterations on each temporal level over the first 6 XBraid iterations. Each δt (column) value corresponds to a temporal level, while the rows represent different XBraid iterations.

It is clear that the work required to complete a Newton solve varies with time step size, but how this affects MGRIT efficiency requires some more discussion. One might initially guess that the increased Newton iteration counts on the coarse grids are irrelevant, after all, for this example we took around 900 Newton iterations on the coarse grid overall, compared to 200 000 on the fine grid. However when using a large numbers of processors, each owning only a few time steps, the cost of the coarse grid solves becomes pivotal.

Consider the case where each MPI process owns m points in time on the finest level, i.e., one CF-interval. In [10] it was shown that this was an efficient decomposition. On coarser levels, each process will then own at most one point in time. Table 3 gives, for the processes active on each temporal level, the estimated average number of Newton iterations to carry out one FCF-relaxation. On the finest grid, these numbers are $2m$ times the values in Table 2 because FCF-relaxation (for larger m) involves roughly $2m$ time step evaluations. Then on coarser grids, the numbers are simply twice what is in Table 2 because each processor owns at most one point, and all fine points are relaxed twice.

In this highly parallel setting, it is immediately clear that the large number of Newton iterations on coarse levels will dominate the cost of a V-cycle. The dominant cost of a V-cycle is relaxation, which is done at each level, meaning that the final column in Table 3 is a cost estimate for each V-cycle. In contrast for the linear setting (when using an optimal spatial solver), the work required would be independent of time step size. For instance, the last row of the table would read similar to $6.88m$, 6.88, 6.88, ... for a total of 61.9. When targeting an MGRIT efficiency similar to a linear problem, this will be a key issue. However, achieving an a_ℓ for a given δt, δx combination that is smaller than the baseline numbers in Table 1 is difficult.

While the metric we show is a_ℓ, another parallel performance issue is the variance in the number of Newton iterations per time step. MGRIT behavior mirrors Table 1, with the max and min being roughly a factor of 2 or 3 apart on the coarser levels. On these coarser grids where at most one time step is given to each processor, synchronization effects imply that an F- or C-relaxation cannot complete until the slowest processor finishes. We choose to show a_ℓ for brevity and note that its value also tracks that of the maximum number of Newton iterations per time step. A decrease in a_ℓ generally corresponds to a decrease in the max. We draw the readers attention to this, because this behavior is problem dependent.

In conclusion, our goal is to minimize the number of Newton iterations per time step on the coarse grids. Since Newton iterations are the main computational kernel of each Φ application, we expect this to provide needed improvement in MGRIT performance. By itself, this naive application of MGRIT scales very poorly when compared to the comparable linear problem (see sections 6.1 and 6.2).

5 Efficient MGRIT for the model nonlinear problem

In this section, we investigate a variety approaches designed to make MGRIT efficient for our chosen model nonlinear parabolic problem (3). The goal is to achieve a similar efficiency as that seen for a corresponding linear problem. We seek improvements over the naive MGRIT results from Section 4.

5.1 Solver ID table

Given the number of options considered, we now give each solver a numerical id to make the discussion easier. Table 4 presents each solver considered and it's runtime for the chosen test problem size. Each solver option is discussed in more detail in the following subsections. However, we do provide a brief description here for the reader's convenience.

Solver id 0 refers to the naive MGRIT approach from Section 4.3. The column heading "Spatial Grids" refers to the number of spatial grids used and is the option introduced in Section 5.2. A value of 1 for "Spatial Grids" indicates no spatial coarsening, while 4 means that the finest spatial grid is coarsened 3 times for a total of 4 grids. The finest grid is 64×64, so coarsening further in space is not possible. The difference between solvers 1 and 2 is that solver 1 delays spatial coarsening so that it begins on the third coarse grid. Solver 2 begins spatial coarsening immediately on the first coarse grid. We note that there are only 6 grids for 4096 time steps and $m = 4$. Given the bad effects on convergence visible from "no delay" in solver 2, unless otherwise mentioned, we always delay spatial coarsening. See Section 5.2 for more details.

Solver id's 3, 4, 5 and 6 correspond to the improved initial guess introduced in Section 5.3. Here, "PMI" means that the previous MGRIT iteration is used as the initial guess to each Newton solve. This happens either at every point, or only at C-points, depending on whether all points or only C-points are stored.

Solver id's 7 and 8 correspond to turning on the "Skip" option introduced in Section 5.4. There, the concept of skipping unnecessary work during the first MGRIT down cycle during iteration 0 is introduced.

Solver id's 9 and 10 correspond to having a "fixed" or "scaled" Newton tolerance on coarse grids, as introduced in Section 5.5. Essentially, the Newton tolerance is either fixed on all levels, or relaxed on coarse grids.

Solver id's 11 and 12 correspond to having "Cheap first three iterations" as introduced in Section 5.6. Here, we relax further the Newton tolerance during the first three MGRIT iterations.

Solver id's 13, 14, 15 and 16 reduce the number of levels in the hierarchy by setting larger "Coarsest grid sizes" as introduced in Section 5.7. For instance with $m = 4$, using a coarsest grid size of 16 instead of 4 removes the coarsest level in the hierarchy. This can improve the time-to-solution by avoiding cycling between very small grids.

5.2 MGRIT with spatial coarsening

We now consider updating the naive solver from Section 4 with the spatial coarsening in Algorithm 1. In general, the user's code defines the separate spatial interpolation and restriction functions $P_x()$ and $R_x()$. We choose the natural finite element restriction operator (and its transpose) to move between regularly refined grids. This operator is provided by MFEM. Interpolation is the scaled (by $1/4$) transpose of restriction so that $RP \approx I$. The choice of spatial interpolation operators is an area of active research; however, it is not

Solver id	Spatial grids	PMI	Skip	Newton Tol	Cheap first three iters	Coarsest grid size	Runtime	Num iters	Runtime per iter
0	1	Never	no	fixed	no	4	$1898s$	9	$211s$
1	4	Never	no	fixed	no	4	$1215s$	9	$135s$
2	4 no delay	Never	no	fixed	no	4	$2577s$	35	$74s$
3	1	C points	no	fixed	no	4	$1278s$	9	$142s$
4	4	C points	no	fixed	no	4	$953s$	9	$106s$
5	1	Always	no	fixed	no	4	$1193s$	9	$132s$
6	4	Always	no	fixed	no	4	$864s$	9	$96s$
7	1	Always	yes	fixed	no	4	$826s$	8	$103s$
8	4	Always	yes	fixed	no	4	$708s$	9	$79s$
9	1	Always	yes	scaled	no	4	$746s$	8	$93s$
10	4	Always	yes	scaled	no	4	$681s$	9	$76s$
11	1	Always	yes	scaled	yes	4	$636s$	8	$80s$
12	4	Always	yes	scaled	yes	4	$574s$	9	$64s$
13	1	Always	yes	scaled	yes	16	$682s$	8	$85s$
14	4	Always	yes	scaled	yes	16	$536s$	9	$60s$
15	1	Always	yes	scaled	yes	64	$890s$	8	$111s$
16	4	Always	yes	scaled	yes	64	$668s$	9	$74s$

Table 4: Overall runtimes, iteration counts and average time per iteration for our various solver options, with a $(64)^2 \times 4096$ space-time grid.

surprising that using a scaling so that RP resembles an oblique projection helps MGRIT convergence. We note that a better choice here could lead to improved results below.

We are interested in spatial coarsening because it will make the Newton iterations both better conditioned and over a smaller spatial grid (i.e., cheaper). From a high-level, δt controls the conditioning of a backward Euler time step,

$$\left(I - \frac{\delta t}{\delta x^2} A(u_{k+1})\right) = f(u_k),$$

where A is some nonlinear diffusion integrator. As $\delta t/\delta x^2$ increases the operator moves away from the identity, becoming more expensive to solve. By coarsening in space as well as time we can control this ratio, making the nonlinear inversion of Φ cheaper on the coarse grid.

One benefit of MGRIT is that it is that if we get the exact solution on the coarse grid, then our interpolation (injection and F-relaxation) yields the exact solution on fine grid. Error introduced by restriction and interpolation between spatial meshes removes this property. Without this exactness, any error modes in the null space of the restriction operator, must be damped solely by FCF-relaxation. In many cases this causes degradation of MGRIT convergence. More precisely, with spatial restriction and prolongation, the two grid error propagation operator from (13) becomes

$$P(I - P_x B_\Delta^{-1} R_x A_\Delta)(I - A_\Delta) R_I. \tag{25}$$

Table 5 explores this idea by comparing a_ℓ on each temporal level and iteration. The values for a_ℓ increase on temporal level in Table 5, but considerably less than in Table 2. However, these values for a_ℓ are still higher than those for the same δt and δx in Table 1, thus leaving room for further improvement.

Table 4 verifies our heuristic and shows the effect of spatial coarsening on MGRIT runtimes. Here, we focus on solver id's 0, 1 and 2 (see Section 5.1). Solver 1, where spatial coarsening begins on the first coarse grid, leads to a degradation in MGRIT convergence and is still a subject of active research and likely relates to the discussion above regarding the scaling of interpolation and the information lost when moving to coarse levels as described in equation (25). For practical purposes, this solver is unusable as the convergence

Iteration	$\delta t = 1/1024$	1/256	1/64	1/16	1/4	1
0	6.67	6.72	8.67	14.4	15.5	11.8
1	3.70	4.31	8.39	11.9	13.6	12.2
2	3.48	4.13	8.36	11.3	13.3	11.8
3	3.45	4.06	8.49	11.1	13.3	11.8
4	3.45	4.06	8.42	11.1	13.4	11.8
5	3.45	4.06	8.44	11.1	13.4	11.8

Table 5: The average number of Newton iterations per time step (a_ℓ) with spatial coarsening across each temporal level and MGRIT iteration. Spatial coarsening begins on the fourth time level, $\delta t = 0.0625$. c.f. table 2

degradation continues for larger problems. This is unfortunate given that this approach has a much smaller runtime per iteration. On the other hand, Solver 2, where spatial coarsening is delayed until the third coarse grid provides no degradation in convergence and an improved time to solution. This is the strategy pursued in this paper and it has proved to be robust in our tests.

5.3 MGRIT with an improved initial guess for Newton's method

We now consider updating the solver from the last section with a better initial guess for the Newton solver based on the previous MGRIT iteration. Classically, the best available initial guess for the Newton solve is the previous time step. However, after MGRIT has completed one iteration we have two choices, the previous time step and the solution from the previous MGRIT iteration. As MGRIT converges the solution from the previous iteration becomes an ever improving initial guess. As an implementation note, after some MGRIT iterations, the initial guess to Newton can become so good that it satisfies the Newton solver's tolerance. However, MGRIT needs relaxation to update the solution, or face stagnation. Therefore, we always force the Newton solver to iterate at least until the nonlinear residual is reduced.

The main drawback of using the previous MGRIT iteration is that it requires us to store the solution at all points in time. This is a large memory limitation not present when using the previous time step, where we store the solution at only the coarse points. One efficiency/memory compromise is to use the previous MGRIT iteration as the initial guess *only at* coarse points, and the previous time step at all fine points.

Tables 6a and 6b show our heuristic a_ℓ using the PMI as the initial guess for the cases of no spatial coarsening and with spatial coarsening, respectively. Compared to Table 5, we see large reductions in a_ℓ, but primarily at the finer levels. Compared to Table 2, we see a large reduction in a_ℓ across all temporal grids. Furthermore, we see reductions during the later MGRIT iterations. By iteration three we see a_ℓ across all grids comparable with those from the sequential solver in Table 1, both with and without spatial coarsening.

Table 4 validates the heuristic and shows reduced runtimes for MGRIT using the PMI as the initial guess for the Newton solver at all points and only at coarse points (C-points). Here, we focus on solver id's 3, 4, 5 and 6 (see Section 5.1). The improved a_ℓ from Table 6 provides for improved runtimes.

In conclusion, these results indicate that using the PMI as the initial guess after the first MGRIT iteration is very beneficial. During the first MGRIT iteration, no value but the previous time step exists, and thus, must be used as the initial guess. For users with more memory limitations, using the PMI only at coarse points is a good option.

5.4 Skipping Unnecessary Work

Even with the improvements so far, a_ℓ remains large during the first three MGRIT iterations. Consider iteration 0 and the down-cycle and up-cycle parts of Figure 4. In our case, where no prior knowledge of the solution is available, it is clear that relaxation during the down cycle of iteration 0 provides little to no benefit, because no global information has yet propagated from the initial condition. The coarse-grid solve and following up-cycle during iteration 0 is the first time that global information is propagated. Therefore, we consider doing no relaxation or work of any kind during the down-cycle during iteration 0. The solution

Iteration	$\delta t = 1/1024$	1/256	1/64	1/16	1/4	1
0	8.87	8.29	8.45	10.1	13.2	14.0
1	4.67	3.89	4.78	6.43	7.88	11.5
2	3.39	3.10	3.40	5.68	8.18	11.0
3	2.52	2.12	2.89	5.37	8.28	11.5
4	2.02	2.00	2.86	5.33	8.48	11.2
5	2.00	2.00	2.86	5.32	8.30	11.2

(a) Previous MGRIT iteration as initial guess

Iteration	$\delta t = 1/1024$	1/256	1/64	1/16	1/4	1
0	8.87	8.31	8.49	14.2	10.9	8.00
1	4.75	4.04	4.92	6.04	6.63	8.00
2	3.47	3.20	3.47	5.38	6.60	8.00
3	2.55	2.11	2.90	5.36	6.53	8.00
4	2.01	2.00	2.86	5.38	6.53	8.00
5	2.00	2.00	2.86	5.36	6.53	8.00

(b) Previous MGRIT iteration at all points with spatial coarsening (4 levels)

Table 6: Comparison of the average number of Newton iterations per time step (a_ℓ), across each temporal level and MGRIT iteration

on the finest-grid is injected to the coarsest-grid and then serially propagated there. Then, the solution is interpolated back to the finest-grid. This strategy is most similar to full multigrid methods.

Table 7 shows a_ℓ for this approach, and the reader will note that these numbers are not reduced compared to the last solver modification in Table 6, but rather a_ℓ has increased substantially during the first iteration. However, the corresponding solver id's 7 and 8 in Table 4 show an overall gain in runtime. This is due to the fact that the time-stepping routine is being called far fewer times during iteration 0. For instance, the "0" denotes the fact that there is no work done on the first level for iteration 0, by far the most expensive level. Moreover, the work on the coarse levels during iteration 0 represent only the Newton iterations required to solve sequentially on the coarsest grid and then interpolate values to finer grids. Remember, there is no relaxation done during the down-cycle during iteration 0.

5.5 Newton solver accuracy

We now consider updating the solver from the last section by loosening the Newton solver tolerance on coarse grids. Reduction in the accuracy of the coarse grid solves proved to be an effective way of minimizing overall runtimes for linear problems [10]. In that case the number of spatial V cycles allowed by the coarse grid linear solver was capped. Here, we investigate varying the solver tolerance across temporal levels, rather than capping iteration counts (see Remark 5.1).

One way to reduce work on the coarse grids is to loosen the Newton solver tolerance on a per-level basis, making coarse grids have a looser value. We scale the Newton solver tolerance on the coarse grid as

$$tol = \min(4^\ell \; tol_f, 0.001), \tag{26}$$

where $\ell = 0$ is the finest level, and tol_f is the desired Newton tolerance on the fine grid, in this case, $tol_f = 1 \times 10^{-7}$. This choice was based on the fact that $m = 4$, followed by some experimentation. In addition to this when spatial coarsening is used, the nonlinear residual on the coarse spatial grids in the Newton solver is scaled by the relative increase in the grid spacing. This allows the residual norm on a spatially coarsened grid to be compared in an "apples-to-apples" way to a norm on a finer grid.

Table 8 shows our heuristic. While we see improvement compared to Table 7, the issue still remains where time steps on coarse levels are more difficult to solve. Further investigation in Table 4 (solver id's 9

Iteration	$\delta t = 1/1024$	1/256	1/64	1/16	1/4	1
0	0	18.8	23.0	26.3	30.2	29.8
1	2.71	2.93	3.40	5.69	8.70	11.0
2	2.51	2.16	3.00	5.45	8.30	11.5
3	2.06	2.01	2.86	5.33	8.45	11.5
4	2.00	2.00	2.86	5.32	8.37	11.2
5	2.00	2.00	2.86	5.32	8.37	11.2
6	2.00	2.00	2.86	5.32	8.37	11.2
7	2.00	2.00	2.86	5.32	8.37	11.2

(a) Skipping unnecessary work

Iteration	$\delta t = 1/1024$	1/256	1/64	1/16	1/4	1
0	0	18.8	23.0	22.1	22.1	21.2
1	2.73	3.09	3.74	18.1	16.7	15.0
2	2.96	2.99	3.74	1.01	8.78	8.75
3	2.67	2.41	3.11	5.60	6.58	7.75
4	2.20	2.04	2.88	5.36	6.53	8.00
5	2.01	2.00	2.86	5.36	6.53	8.00
6	2.00	2.00	2.86	5.36	6.53	8.00
7	2.00	2.00	2.86	5.36	6.53	8.00
8	2.00	2.00	2.86	5.36	6.53	8.00

(b) Skipping unnecessary work with spatial coarsening (4 levels)

Table 7: Comparison of the average number of Newton iterations per time step (a_ℓ), across each temporal level and MGRIT iteration, when skipping work during the MGRIT down-cycle during iteration 0.

and 10) shows that our heuristic is again accurate. The reduction of a_ℓ on the coarse levels does result in a speedup.

Remark 5.1 *An alternative to the variable tolerance, is to simply cap the number of Newton iterations on the coarse-grids in a manner similar to that pursued for linear problems in [10]. With this strategy, the number of Newton iterations is allowed to vary on the fine-grid, until the tolerance is met. But on coarse-grids, the iteration count is capped. However, this is more dangerous in the nonlinear setting because each Newton iteration is not guaranteed to reduce the residual by a fixed amount, as opposed to the linear case in [10]. Indeed, our experiments have shown that this approach easily leads to degraded MGRIT convergence and must be tuned to each individual grid size, i.e., our weak scaling studies show an increasing iteration count for this approach. We therefore do not consider it any further.*

5.6 Cheap initial iterates

The initial three iterates are the most expensive, with a_ℓ significantly higher on all levels. The solution at this point is still inaccurate, so another obvious modification is to reduce the Newton tolerance during the first two iterations. Our simple strategy here is to use 10^{-3} (as opposed to 10^{-7}) for the Newton tolerance strategy in equation (26) during the first three iterations, followed by a return to (26). Table 9 shows our heuristic indicating a significant drop in a_ℓ during iterations 0, 1 and 2 when compared to the previous solver modification in Table 8. Table 4 (solver id's 11 and 12) validates this by showing a decreased runtime and no degradation in overall MGRIT convergence.

Remark 5.2 *One other solver modification related to solver performance that we use is capping the number of inner linear iterations on coarse grids, similar to [10]. Here, the BoomerAMG solver is used for each Newton iteration and we cap the number of BoomerAMG iterations at 8. This number can be lowered still (our experiments went as low as 5 with no noticeable degradation), but to avoid over-tuning the problem, we*

Iteration	$\delta t = 1/1024$	1/256	1/64	1/16	1/4	1
0	0	18.7	22.7	25.7	29.6	29.2
1	2.71	2.71	3.01	4.47	7.33	9.75
2	2.51	2.08	2.56	4.35	6.98	9.75
3	2.06	2.00	2.43	4.22	7.05	10.0
4	2.00	2.00	2.43	4.22	7.00	10.0
5	2.00	2.00	2.43	4.22	7.00	10.0

(a) Level-based Newton tolerance

Iteration	$\delta t = 1/1024$	1/256	1/64	1/16	1/4	1
0	0	18.7	22.7	21.8	21.4	20.5
1	2.73	2.88	3.36	17.7	16.0	14.0
2	2.96	2.80	3.29	9.57	7.90	7.75
3	2.67	2.27	2.68	5.07	5.65	7.00
4	2.20	2.01	2.44	4.96	5.72	7.25
5	2.01	2.00	2.43	4.96	5.72	7.25

(b) Level-based Newton tolerance with spatial coarsening (4 levels)

Table 8: Average number of Newton iterations per time step (a_ℓ) across each temporal level and MGRIT iteration.

Iteration	$\delta t = 1/1024$	1/256	1/64	1/16	1/4	1
0	0	17.6	21.6	24.9	29.1	28.5
1	2.00	2.02	2.02	2.40	5.73	9.50
2	2.01	2.00	2.00	2.30	5.12	8.25
3	2.11	2.01	2.45	4.22	7.05	10.0
4	2.00	2.00	2.45	4.22	7.00	10.0
5	2.00	2.00	2.45	4.22	7.00	10.0

(a) Cheap initial iterates

Iteration	$\delta t = 1/1024$	1/256	1/64	1/16	1/4	1
0	0	17.6	21.6	20.8	20.8	19.8
1	2.02	2.10	2.21	16.7	15.1	14.0
2	2.13	2.11	2.13	8.21	7.00	7.75
3	2.75	2.29	2.70	5.07	5.65	7.00
4	2.21	2.01	2.46	4.96	5.72	7.25
5	2.01	2.00	2.45	4.96	5.72	7.25

(b) Cheap initial iterates with spatial coarsening (4 levels)

Table 9: Average number of Newton iterations per time step (a_ℓ) across each temporal level and MGRIT iteration.

leave it at 8. When we refer to solvers 11 and 12 (and all higher numbers) we have added this feature. Given it's small impact, we did not devote an entire solver id to this change.

5.7 Setting the coarsest grid size

The final algorithmic enhancement considered is the size of the coarsest grid. Given how relatively expensive the Newton solver is on coarse grids, the question naturally arises, whether truncating the number of levels in the hierarchy can be beneficial. However, this involves a trade-off. When a level is truncated from the hierarchy, the expensive Newton solves are no longer done there and the communication involved with visiting that level is avoided. However, the sequential part of the algorithm increases because the coarsest grid size has now increased. The best coarsest grid size is therefore naturally problem dependent. So far, we have used a coarsest grid size of 4. We now consider 16 and 64 as well.

The average number of Newton iterations is very similar to Table 9 except the coarsest level is removed when using a coarsest grid size of 16, and that the coarsest two levels are removed for a coarsest grid size of 64. The case of 16 (solver id's 13 and 14 in Table 4) show a significant speedup of $44s$ when using spatial coarsening but a slow down of $19s$ for the case of no spatial coarsening. This is because spatial coarsening makes the spatial grid on the coarsest grid much smaller, and hence the sequential solve required on the coarsest level is much cheaper. In other words the penalty for a larger coarsest grid size is much smaller for the case of spatial coarsening, thus allowing for the benefits to be visible.

For the case of a coarsest grid size of 64 (solver id's 15 and 16 in Table 4) the extra work from the larger sequential component of the solve on the coarsest level swamps any benefit and the runtimes increase substantially in both cases. Given that the solver id 14 is the best overall performing solver, we will choose a maximum coarse grid size of 16 in our experiments.

5.8 Most effective improvements

While we have outlined many improvements, Table 4 makes it clear that two of the improvements (spatial coarsening and the improved initial guess) are the most important. This can be seen by comparing solver 0 to solver 1 to see the impact of spatial coarsening, a 36% improvement. Then, when solver 1 and solver 6 are compared to see the impact of the improved initial guess, a further 29% improvement is seen. The other four strategies in concert combine for another 48% improvement (compare solver 6 with solver 14). During the subsequent scaling studies we therefore focus on solvers 0, 1, 6 and 14.

6 Scaling Studies

Previous sections have focused on producing the most efficient Newton solver as a proxy for MGRIT efficiency. We now carry out the parallel scaling studies to validate that heuristic.

6.1 Optimal multigrid scaling

We consider a domain refinement study to test the optimality of MGRIT for this problem, in a manner similar to how spatial multigrid optimality is tested experimentally. We fix the space-time domain to $([0,2]^2 \times [0,4])$ and scale up the spatial and temporal resolution, keeping $\delta t/\delta x^2$ fixed. For runs using spatial coarsening, the number of levels of spatial coarsening was increased on each subsequent test, resulting in 4 levels of spatial coarsening on the largest space-time grid of $256^2 \times 65\,536$. The solver id's considered are 0, 1, 6, 14, so that we can examine (respectively) the effects of spatial coarsening, the improved initial guess for the Newton solver, followed by the other enhancements.

Results are shown in Table 10. In general optimal iteration counts are observed, bounded independently of problem size for all the considered solver options. Unfortunately, using this experiment for weak scaling timings requires more processors than our machine provides (131K processors are available). Consider a weak scaling experiment where the smallest problem size involves 8 compute nodes, with 16 processors per node, for a total of 128 processors. We need a base test case that involves some off-node communication, hence the choice of 8 compute nodes. To maintain a constant problem size per node and a constant $\delta t/\delta x^2$, we must quadruple the number of time points as the spatial problem size is doubled. This corresponds to

Solver ID / $N_x^2 \times N_t$	$32^2 \times 256$	$32^2 \times 1\,024$	$64^2 \times 4\,096$	$128^2 \times 16\,384$	$256^2 \times 65\,536$
0	4	7	9	10	10
1	6	8	9	11	10
6	6	8	9	11	10
14	6	8	9	11	10

(a) Iterations

Table 10: Weak scaling study

increasing the node count by a factor of 16. Thus to obtain four data points, $128 * 16^3 = 524\,288$ processors would be required.

The iteration counts bounce from 9 to 11 and then back to 10 for the last three problem sizes. This is primarily because the stopping tolerance is barely met by the 9 iteration case, barely missed after 10 iterations for the 11 iteration case, and then barely met again in the 10 iteration case.

6.2 Strong scaling

Both MGRIT and sequential time stepping are $O(N)$ optimal, but the constant for MGRIT is larger, leading to a crossover point. To illustrate this, we do a strong scaling study of MGRIT for the space-time grid of $(128)^2 \times 16384$. Figures 5a and 5b show the results. The plot for "id=14, linear problem" corresponds to the comparable linear problem of $p = 2$ in equation (3). This allows us to compare MGRIT's scaling for the nonlinear problem with the scaling for the corresponding linear problem. To generate the date for "id=14, linear problem", we simply set $p = 2$ in the code and make no other optimizations, e.g., the spatial matrix and solver are still built every Φ application as is required in the nonlinear case, so that the comparison is fair. We note that in [10] the spatial matrix and solver need only be built once for the constant coefficient heat equation.

The other plots are for the sequential ("Time-stepping") time-stepping code and solver id's 0, 1, 6, 12 and 14. This allows for a comparison of naive MGRIT (id=0) with the effects of spatial coarsening (id=1), the improved initial guess (id=6) and the effects of all the other improvements (ids=12, 14).

We show the results for 16 processors in space and $m = 16$ on the left in Figure 5, while the right shows that for 32 processors in space and $m = 4$. We change the coarsening factor m to indicate that this parameter can be changed, depending on the number of overall processors available to maximize speedup. For the $m = 16$ case, the crossover point at which MGRIT is beneficial well below 1024 processors, or about 50 to 60 processors in time. The maximum speedup attained is a factor of 6.7 at 16384 processors. For the $m = 4$ case, the crossover point is at about 2000 processors, or about 500 processors in time. Yet, this smaller coarsening factor allows you to use more of the machine, and at 130K processors, the speedup is 18. The data points in each plot end when there are m points in time per processor, i.e., when there is no more benefit to using more processors in time. This is because FCF-relaxation is sequential over each interval of m points.

Figure 5 highlights another important aspect of the MGRIT algorithm. Increasing processors in space allows for greater scaling potential, but, given a fixed number of processors, it is often beneficial to bias the processor distribution towards temporal processors until m equals the number of time points per processor. A last important point is that MGRIT converged in 9 iterations for the tests here, but discretization error is reached after 5 iterations. Thus, timings presented here are largely understated. Halting MGRIT in a more automatic fashion after discretization error is reached is a topic of further research.

Compared to the superior strong scaling in Figure 1, these results are not as good. Essentially, we desire our lines to be straighter. However, achieving this is difficult. The dataset for the linear heat equation ("id=12, linear problem") is very similar to the experiment in Figure 1, except that (1) linear finite elements on a regular grid are used in space, as opposed to finite differencing, (2) the BoomerAMG solver in hypre is used, as opposed to the more efficient geometric-algebraic solver PFMG in hypre and (3) the spatial discretization and spatial multigrid solver must be built every time step. However, this dataset exhibits the same strong scaling issues, i.e., it does not look (qualitatively) like the plots in Figure 1. Improving the scaling of MGRIT here will require addressing these differences, with (2) and (3) the more likely culprits for

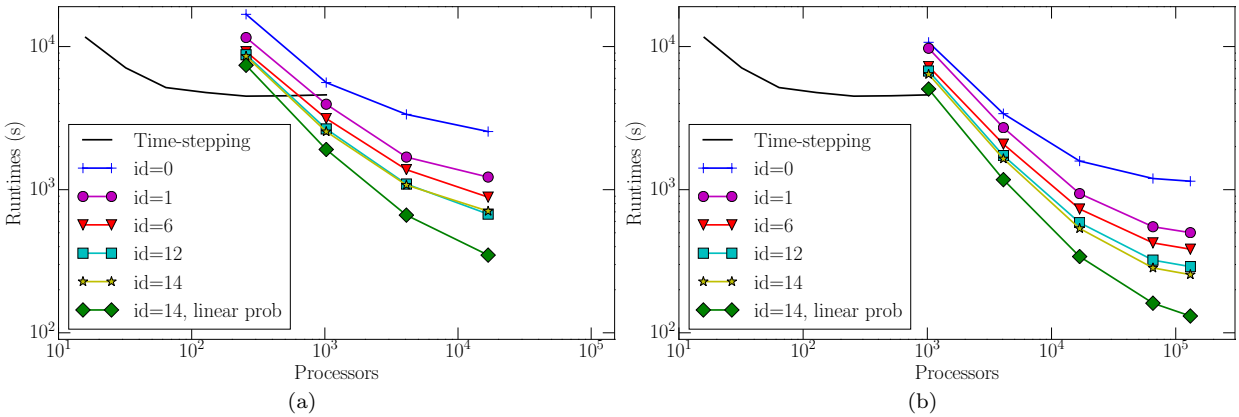

Figure 5: Strong scaling study for a $(128)^2 \times 16385$ space-time grid, Left: 16 processors in space, $m = 16$, Right: 32 processors in space, $m = 4$.

reduced scalability. We discount the difference (1) because it does not change the sparsity pattern of the spatial operator, nor does it qualitatively change the convergence rate of the spatial multigrid solver. Thus, we believe that the most likely culprits for the strong scaling degradation are the parallel finite-element matrix assembly and the BoomerAMG setup-phase, although we note that BoomerAMG is known to be an efficient spatial multigrid code. Remedying issues such as these is beyond the scope of this paper.

Last, we compare MGRIT for the nonlinear problem and comparable linear problem. The overall scaling behavior is similar, with a deterioration for both at larger processor counts. However, MGRIT for the nonlinear problem does scale somewhat more poorly. This is due in large part to the phenomenon discussed with Table 3, where the relatively more expensive coarse grids for the nonlinear problem reduce scalability. To reiterate, the addition of more processors does not change the cost of coarser levels, where there is one time point per processor. Here, the nonlinear time steps are many times more expensive than the corresponding linear ones and as illustrated by Table 3 these coarse levels (which cannot be subdivided by more processors in time) quickly become the dominate cost of a V-cycle. The addition of more processors only reduces the time taken on the finer (and eventually only the finest) level. The chief strategy under research now is to improve spatial coarsening so that it can begin on the first coarse grid and significantly reduce this effect.

It is important, however, that even at it's worst, MGRIT for the nonlinear problem is about 3 times slower than for the linear problem. This is a good result, considering MGRIT takes at least 2, but often many more linear solves per time step (depending on level and iteration) as for the linear problem.

7 Conclusions

The MGRIT algorithm effectively adds temporal parallelism to existing sequential solvers and has been shown to be effective for linear problems. However, when moving to the nonlinear setting, the relatively large time-step sizes on coarse grids make the application of MGRIT nontrivial. The proposed measures allow MGRIT to achieve similar performance to a comparable linear problem.

In summary, we found that, after the first iteration, the user should always use the solution from the previous MGRIT iteration as the initial guess to the nonlinear time-stepping routine (here, a Newton solver). When memory constraints inhibit this approach, using the previous solution at only coarse points still provides dramatic speedups. Secondly, spatial coarsening should be used whenever possible. For the linear example in Figure 1, spatial coarsening was implemented on all levels effectively. We have seen that for our nonlinear model problem this was not possible, however spatial coarsening on only the lower time levels provided dramatic speedups through a reduction in Newton iterations and reduced compute times due to the smaller problem sizes. These two changes gave the largest speedup. The other changes, when combined, also effected a large speedup.

Weak scaling results showed that MGRIT is a scalable algorithm for nonlinear problems, with iteration

counts bounded independently of problem size. Strong scaling showed benefits of MGRIT, with 18x speedups seen over the corresponding sequential time stepper. The scaling is not as ideal as in [10], but with the modifications given here, we were able to attain similar scaling when compared to the corresponding linear problem.

7.1 Future work

This topic has been well explored in this this study. The remaining topic to address is the efficiency issues in the MFEM finite-element matrix construction routines and BoomerAMG setup-phase that contribute to the poor strong scaling.

References

[1] P. Bastian, J. Burmeister, and G. Horton. Implementation of a parallel multigrid method for parabolic partial differential equations. In W. Hackbusch, editor, *Parallel Algorithms for PDEs, Proc. 6th GAMM Seminar Kiel, January 19-21, 1990*, pages 18–27, Braunschweig, 1990. Vieweg Verlag.

[2] B. Bjorn and J. Rowlett. Mathematical models for erosion and the optimal transportation of sediment. *International Journal of Nonlinear Sciences and Numerical Simulation*, 14(6):323–337, 2013.

[3] A. Brandt. Multi-level adaptive computations in fluid dynamics, 1979. Technical Report AIAA-79-1455, AIAA, Williamsburg, VA.

[4] A. Brandt, S. F. McCormick, and J. W. Ruge. Algebraic multigrid (AMG) for sparse matrix equations. In D. J. Evans, editor, *Sparsity and Its Applications*, pages 257–284. Cambridge Univ. Press, Cambridge, 1984.

[5] P. Chartier and B. Philippe. A parallel shooting technique for solving dissipative ODEs. *Computing*, 51(3-4):209–236, 1993.

[6] Andrew J. Christlieb, Colin B. Macdonald, and Benjamin W. Ong. Parallel high-order integrators. *SIAM Journal on Scientific Computing*, 32(2):818–835, 2010.

[7] H. De Sterck, T. A. Manteuffel, S. F. McCormick, and L. Olson. Least-squares finite element methods and algebraic multigrid solvers for linear hyperbolic PDEs. *SIAM J. Sci. Comput.*, 26(1):31–54, 2004.

[8] A. Elmoataz, M. Toutain, and D. Tenbrinck. On the p-laplacian and ∞-laplacian on graphs with applications in image and data processing. *SIAM Journal on Imaging Sciences*, 8(4):2412–2451, 2015.

[9] M. Emmett and M. L. Minion. Toward an efficient parallel in time method for partial differential equations. *Commun. Appl. Math. Comput. Sci.*, 7(1):105–132, 2012.

[10] R. D. Falgout, S. Friedhoff, Tz. V. Kolev, S. P. MacLachlan, and J. B. Schroder. Parallel time integration with multigrid. *SIAM Journal on Scientific Computing*, 36(6):C635–C661, 2014.

[11] R. D. Falgout, A. Katz, Tz.V. Kolev, J. B. Schroder, A. Wissink, and U. M. Yang. Parallel time integration with multigrid reduction for a compressible fluid dynamics application. *Journal of Computational Physics*, (submitted), 2015.

[12] M. J. Gander. *50 years of Time Parallel Time Integration*. Multiple Shooting and Time Domain Decomposition. Springer, 2015. In press.

[13] M. J. Gander and S. Vandewalle. Analysis of the parareal time-parallel time-integration method. *SIAM Journal on Scientific Computing*, 29:556–578, 2007.

[14] Stefan Güttel. A parallel overlapping time-domain decomposition method for ODEs. In *Domain decomposition methods in science and engineering XX*, volume 91 of *Lect. Notes Comput. Sci. Eng.*, pages 459–466. Springer, Heidelberg, 2013.

[15] W. Hackbusch. Parabolic multigrid methods. In *Computing methods in applied sciences and engineering, VI (Versailles, 1983)*, pages 189–197. North-Holland, Amsterdam, 1984.

[16] G. Horton. The time-parallel multigrid method. *Comm. Appl. Numer. Methods*, 8(9):585–595, 1992.

[17] G. Horton and R. Knirsch. A time-parallel multigrid-extrapolation method for parabolic partial differential equations. *Parallel Comput.*, 18(1):21–29, 1992.

[18] G. Horton and S. Vandewalle. A space-time multigrid method for parabolic partial differential equations. *SIAM J. Sci. Comput.*, 16(4):848–864, 1995.

[19] G. Horton, S. Vandewalle, and P. Worley. An algorithm with polylog parallel complexity for solving parabolic partial differential equations. *SIAM J. Sci. Comput.*, 16(3):531–541, 1995.

[20] HYPRE: High performance preconditioners. `http://www.llnl.gov/CASC/hypre/`.

[21] H. B. Keller. *Numerical methods for two-point boundary-value problems*. Blaisdell Publishing Co. Ginn and Co., Waltham, Mass.-Toronto, Ont.-London, 1968.

[22] P. Lindqvist. Notes on the p-laplace equation. Technical report, Department of Mathematics, Ohio State University, 2006.

[23] J.-L. Lions, Y. Maday, and G. Turinici. Résolution d'EDP par un schéma en temps "pararéel". *C. R. Acad. Sci. Paris Sér. I Math.*, 332(7):661–668, 2001.

[24] C. Lubich and A. Ostermann. Multigrid dynamic iteration for parabolic equations. *BIT*, 27(2):216–234, 1987.

[25] Yvon Maday and Einar M. Rønquist. Parallelization in time through tensor-product space-time solvers. *Comptes Rendus Mathématique. Académie des Sciences. Paris*, 346(1-2):113–118, 2008.

[26] S. F. McCormick and J. W. Ruge. Multigrid methods for variational problems. *SIAM J. Numer. Anal.*, 19(5):924–929, 1982.

[27] MFEM: Modular finite element methods. `mfem.googlecode.com`.

[28] M. L. Minion and S. A. Williams. Parareal and spectral deferred corrections. In T. E. Simos, editor, *Numerical Analysis and Applied Mathematics*, number 1048 in AIP Conference Proceedings, pages 388–391. AIP, 2008.

[29] Willard L. Miranker and Werner Liniger. Parallel methods for the numerical integration of ordinary differential equations. *Math. Comp.*, 21:303–320, 1967.

[30] J. Nievergelt. Parallel methods for integrating ordinary differential equations. *Comm. ACM*, 7:731–733, 1964.

[31] J. W. Ruge and K. Stüben. Algebraic multigrid (AMG). In S. F. McCormick, editor, *Multigrid Methods*, Frontiers Appl. Math., pages 73–130. SIAM, Philadelphia, 1987.

[32] Dongwoo Sheen, Ian H. Sloan, and Vidar Thomée. A parallel method for time discretization of parabolic equations based on Laplace transformation and quadrature. *IMA Journal of Numerical Analysis*, 23(2):269–299, 2003.

[33] S. Vandewalle and G. Horton. Fourier mode analysis of the multigrid waveform relaxation and time-parallel multigrid methods. *Computing*, 54(4):317–330, 1995.

[34] S. Vandewalle and R. Piessens. Efficient parallel algorithms for solving initial-boundary value and time-periodic parabolic partial differential equations. *SIAM J. Sci. Statist. Comput.*, 13(6):1330–1346, 1992.

[35] S. G. Vandewalle and E. F. Van de Velde. Space-time concurrent multigrid waveform relaxation. *Ann. Numer. Math.*, 1(1-4):347–360, 1994. Scientific computation and differential equations (Auckland, 1993).

[36] T. Weinzierl and T. Köppl. A geometric space-time multigrid algorithm for the heat equation. *Numer. Math. Theory Methods Appl.*, 5(1):110–130, 2012.

[37] XBraid: Parallel multigrid in time. `http://llnl.gov/casc/xbraid`.